

FAULT TOLERANT SYSTEM, FAULT TOLERANT PROCESSING METHOD AND RECORDING MEDIUM FOR FAULT TOLERANT CONTROL PROGRAM

Publication number: JP2001022599 (A)

Publication date: 2001-01-26

Inventor(s): MATSUSHITA KOZO; MINAZU SHINJI; OKAMOTO JIRO +

Applicant(s): FUJITSU LTD +

Classification:

- international: G06F11/20; G06F15/177; G06F9/46; G06F9/48; G06F11/20; G06F15/16; G06F9/46;
(IPC1-7): G06F11/20; G06F15/177; G06F9/46; G06F9/46

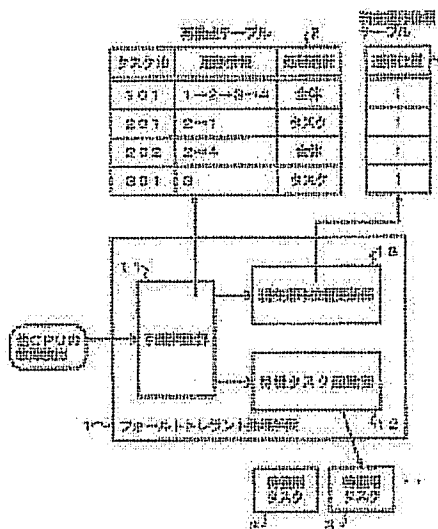
- European:

Application number: JP19990191135 19990706

Priority number(s): JP19990191135 19990706

Abstract of JP 2001022599 (A)

PROBLEM TO BE SOLVED: To easily designate a specific CPU that takes over the processing of another faulty CPU for every task in a multiprocessor system. **SOLUTION:** A take-over decision part 11 searches for a task which is operating with a faulty CPU from the transition information, i.e., the series of the CPU number that takes over the task included in a reconfiguration table 2 and the current transition position showing the CPU under processing in a current transition position table 4 after the CPU fault is detected and then checks the CPU that takes over the task. When the checked CPU is its own one, a standby task start part 12 starts a standby task 3 of the relevant task to take over the processing. A current transition position updating part 13 updates the current transition position of the table 4. When no CPU takes over the task, the task or a total system is discontinued according to the processing selection information.



Data supplied from the *espacenet* database — Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-22599

(P2001-22599A)

(43) 公開日 平成13年1月26日 (2001.1.26)

(51) Int.Cl. ⁷	識別記号	F I	テーマコード* (参考)
G 0 6 F 9/46	3 6 0	C 0 6 F 9/46	3 6 0 B 5 B 0 3 4
	3 3 0		3 3 0 C 5 B 0 4 5
11/20	3 1 0	11/20	3 1 0 F 5 B 0 9 8
15/177	6 7 8	15/177	6 7 8 A

審査請求 未請求 請求項の数6 O L (全 12 頁)

(21) 出願番号 特願平11-191135

(22) 出願日 平成11年7月6日 (1999.7.6)

(71) 出願人 000006223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番
1号

(72) 発明者 松下 耕三

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(72) 発明者 水津 真治

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

(74) 代理人 100087848

弁理士 小笠原 吉義 (外2名)

最終頁に続く

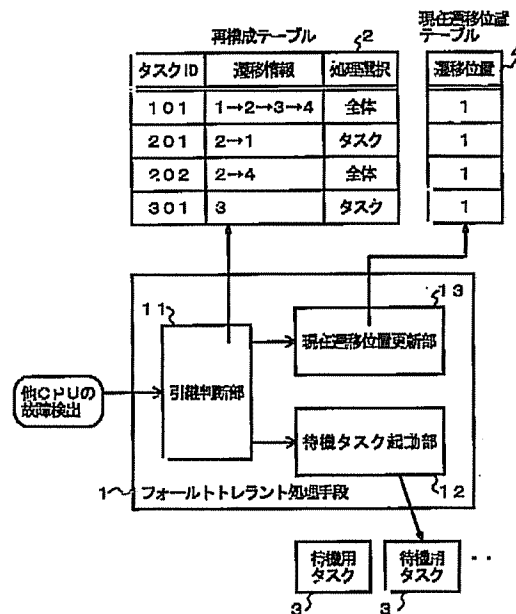
(54) 【発明の名称】 フォールトトレラント・システム、フォールトトレラント処理方法およびフォールトトレラント制御用プログラム記録媒体

(57) 【要約】

【課題】 マルチプロセッサシステムにおいて、あるCPUが故障した場合に、そのCPUの処理を他のCPUがタスク単位で引き継ぐときに、どのCPUが引き継ぐかを簡単に指定することができるようにすることを目的とする。

【解決手段】 引継判断部11は、他のCPUの故障を検出すると、再構成テーブル2中のタスクを引き継ぐCPU番号の系列である遷移情報と、現在遷移位置テーブル4中の現在処理しているCPUを示す現在遷移位置から、故障したCPUで稼働していたタスクを探し、そのタスクを引き継ぐ次のCPUを調べる。引き継ぐCPUが自CPUであれば、待機タスク起動部12は、そのタスクの待機用タスク3を起動し、処理の引き継ぎを行う。現在遷移位置更新部13は、現在遷移位置テーブル4の現在遷移位置を更新する。タスクを引き継ぐCPUがない場合、処理選択情報に基づきタスクまたはシステム全体を停止させる。

本発明のブロック構成図



【特許請求の範囲】

【請求項1】 プロセッサの実行処理単位である複数のタスクが複数のプロセッサに分散して配置され、各プロセッサにおいて処理されるマルチプロセッサシステムにおいて、タスクごとに、プロセッサの故障時にその故障プロセッサで動作していたタスクを引き継ぐプロセッサの系列を定義した遷移情報と、タスクが現在どのプロセッサ上で動作しているのかを示す現在遷移位置とを記憶する記憶手段と、あるプロセッサが故障した場合に、前記遷移情報および前記現在遷移位置の情報にもとづいて、その故障プロセッサで動作していたタスクが自プロセッサで引き継ぐべきタスクであるかどうかを判断する手段と、判断の結果、自プロセッサで引き継ぐものである場合には、そのタスクの引き継ぎを行い、自プロセッサで引き継いだタスクを動作させる手段とを備えることを特徴とするフォールトトレラント・システム。

【請求項2】 前記記憶手段は、タスクを引き継ぐプロセッサがない場合に、そのタスクの停止またはシステム全体の停止のいずれかを示す処理選択情報を記憶し、前記プロセッサは、故障したプロセッサで動作していたタスクを引き継ぐプロセッサがない場合に、前記処理選択情報にもとづいてタスクまたはシステム全体を停止させることを特徴とする請求項1記載のフォールトトレラント・システム。

【請求項3】 プロセッサの実行処理単位である複数のタスクが複数のプロセッサに分散して配置され、各プロセッサにおいて処理されるマルチプロセッサシステムにおけるフォールトトレラント処理方法において、タスクごとに、プロセッサの故障時にその故障プロセッサで動作していたタスクを引き継ぐプロセッサの系列を定義した遷移情報と、タスクが現在どのプロセッサ上で動作しているのかを示す現在遷移位置とを記憶する記憶手段を用い、あるプロセッサが故障した場合に、前記記憶手段を参照し、前記遷移情報および前記現在遷移位置の情報にもとづいて、その故障プロセッサで動作していたタスクが自プロセッサで引き継ぐべきタスクであるかどうかを判断し、判断した結果、自プロセッサで引き継ぐものである場合には、そのタスクの引き継ぎを行い、自プロセッサで引き継いだタスクを動作させることを特徴とするフォールトトレラント処理方法。

【請求項4】 タスクを引き継ぐプロセッサがない場合に、そのタスクの停止またはシステム全体の停止のいずれかを示す処理選択情報を前記記憶手段に記憶しておき、故障したプロセッサで動作していたタスクを引き継ぐプロセッサがない場合に、前記処理選択情報にもとづいてタスクまたはシステム全体を停止させることを特徴とする請求項3記載のフォールトトレラント処理方法。

【請求項5】 マルチプロセッサシステムにおけるフォールトトレラント処理方法をコンピュータによって実現するためのプログラムを記録した記録媒体であって、タ

スクごとに、プロセッサの故障時にその故障プロセッサで動作していたタスクを引き継ぐプロセッサの系列を定義した遷移情報と、タスクが現在どのプロセッサ上で動作しているのかを示す現在遷移位置とを記憶する記憶手段を用い、あるプロセッサが故障した場合に、前記記憶手段を参照し、前記遷移情報および前記現在遷移位置の情報にもとづいて、その故障プロセッサで動作していたタスクが自プロセッサで引き継ぐべきタスクであるかどうかを判断し、判断した結果、自プロセッサで引き継ぐものである場合には、そのタスクの引き継ぎを行い、自プロセッサで引き継いだタスクを動作させる処理を、コンピュータに実行させるプログラムを記録したことを特徴とするフォールトトレラント制御用プログラム記録媒体。

【請求項6】 請求項5記載のフォールトトレラント制御用プログラム記録媒体において、前記プログラムは、前記記憶手段に前記遷移情報とともに記憶された処理選択情報であって、タスクの停止またはシステム全体の停止のいずれかを示す処理選択情報にもとづいて、故障したプロセッサで動作していたタスクを引き継ぐプロセッサがない場合に、タスクまたはシステム全体を停止させる処理を、コンピュータに実行させるプログラムを含むことを特徴とするフォールトトレラント制御用プログラム記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、複数のプロセッサ（以下、CPUという）を搭載したシステムにおけるフォールトトレラント技術に関し、特に、あるCPUが故障した場合に、そのCPUが行っていた処理を他の正常なCPUがタスク単位で引き継ぐとき、どのCPUが引き継ぐのかを正常なCPUが最後の一つになるまで、容易に指定できるようにしたマルチCPUシステムにおけるフォールトトレラント・システム、フォールトトレラント処理方法およびフォールトトレラント制御用プログラム記録媒体に関する。

【0002】

【従来の技術】マルチCPUシステムにおいて、あるCPUが故障したときに、システムを停止することなく、他の正常なCPUによってその処理を継続できるようにするため、例えば、以下のようなフォールトトレラント技術がある。

【0003】(1)「相互ホットスタンバイシステム待機系選択方式（特開平9-81409号公報）」は、電子計算機の障害により稼働系処理機能に対応付けられた待機系処理機能が存在しなくなったとき、操作員の介入なしに自動的に対応付け可能な待機系処理機能を選択して対応付け、ホットスタンバイ関係を構築するためのものであり、自電子計算機で動作する稼働系処理機能の識別子、稼働系処理機能とホットスタンバイ関係を構築す

るように対応付けられた待機系処理機能の識別子および該待機系処理機能が動作する電子計算機の番号からなる情報と、自電子計算機で動作する待機系処理機能の識別子、待機系処理機能とホットスタンバイ関係を構築するように対応付けられた稼働系処理機能の識別子および該稼働系処理機能が動作する電子計算機の番号からなる情報とを登録する処理機能管理表を備えるとともに、電子計算機の障害発生時に、対応付けられた待機系処理機能がなくなった稼働系処理機能からの要求に応じて、前記処理機能管理表を参照して他の電子計算機に対応付けられた稼働系処理機能のない待機系処理機能を選択して要求元の稼働系処理機能に対応付け、自動的にホットスタンバイ関係を構築させる処理機能管理手段とを備える。

【0004】(2)また、「業務引継システム(特願平9-528794号)」は、ホットスタンバイの形態やロードシェアの形態で、複数の処理装置により業務処理の実行を行うシステムにおいて、特に多大なプログラミングを必要とせずに障害発生時の効率的な引き継ぎができるような業務引継システムを提供することを目的として、各処理装置が業務についての現用系であるか待機系であるかを表すテーブルを保持し、業務についての現用系である処理装置の障害発生時に、該障害が発生した処理装置における業務に係る処理を、前記テーブルを参照して当該業務の待機系となる処理装置に引き継ぐようにするものである。

【0005】

【発明が解決しようとする課題】しかし、従来、多数のCPUが次々に故障し続けたとき、故障が生じたCPUの処理をどのCPUが引き継ぐのかを容易に指定する方法はなかった。

【0006】また、上記(1)の技術では、処理機能管理表により、稼働系処理識別子と待機系識別子を管理する必要がある。また、この技術では、処理を引き継ぐ計算機が動的に定まるので、事前に処理を引き継ぐCPUを最後の1台まで静的に定めることができず、さらに、すべてのCPUが同等なハードウェアなどの機能を有している必要があるという問題があった。

【0007】また、上記(2)の技術では、全クラスタのテーブルを作成する必要がある、全クラスタが故障するまで引き継がせたい業務があれば、すべてのテーブルに待機業務を記さなければならない。また、処理を引き継ぐクラスタが待機系を用意している中のどのクラスタになるかは不明であり、優先度を付けてクラスタに処理を引き継がせるような設定をすることができないという問題があった。

【0008】本発明は上記問題点の解決を図り、マルチCPUシステムにおいて、あるCPUが故障した場合に、そのCPUの処理をタスク単位で引き継ぐとき、どのCPUが引き継ぐのかを正常なCPUが最後の1台になるまで、容易に指定することができる手段を提供する

ことを目的とする。

【0009】

【課題を解決するための手段】本発明は、複数のCPUを搭載したシステムにおいて、各CPUは、故障したCPUで動作していたタスクをどのCPUによって引き継ぐのかを、タスクごとにCPU番号の系列によって記した情報を持つ再構成テーブルを持つ。故障したCPUを発見した場合、それを発見したCPUは、他の全CPUに対し故障したCPUのCPU番号(各CPUを識別するための識別番号)を知らせる。各CPUは、再構成テーブルをもとに、故障したCPU上で動作していたタスクを知り、そのタスクを引き継ぐCPUが自分であるかどうかを判断する。自CPUが引き継ぐように指定されているタスクであった場合、そのタスクを引き継ぐために待機していたタスクを起動させる。

【0010】この引き継ぎのため、再構成テーブルには、タスクがどのCPUによって引き継がれていくのかをCPU番号の羅列で表現するだけでよいので、容易に引き継いでいくCPUを指定することができる。また、現在どのCPU上でタスクが実行されているのかを覚えておくことにより、簡単に引き継ぐCPUがどのCPUであるかを調べることができる。

【0011】また、再構成テーブルには、引き継ぐCPUが存在しなくなった場合に、このタスクだけを停止させるのか、またはシステム全体として停止させるのかを指定する情報を持たせておく。これによって、タスクだけの停止、システム全体の停止を選択することができる。

【0012】なお、ここでいうCPU(central processing unit)には、MPU(microprocessing unit)も含まれる。また、ここでいうタスクは、CPUで実行される処理の単位を意味し、プロセスと呼ばれるような処理単位も含まれる広い概念のものである。

【0013】以上の各処理手段をコンピュータによって実現するためのプログラムは、コンピュータが読み取り可能な可搬媒体メモリ、半導体メモリ、ハードディスクなどの適当な記録媒体に格納することができる。

【0014】

【発明の実施の形態】図1は、本発明の概要を示すブロック構成図である。本実施の形態では、マルチCPUシステムにおいて、CPUごとに、フォールトトレラント処理手段1および再構成テーブル2を備える。また、どのCPUもアクセスすることができる記憶領域に、現在遷移位置テーブル4を備える。

【0015】再構成テーブル2は、タスクごとに、各タスクを識別するタスクID(識別子)と、故障が生じたCPU上で実行されていたタスクを、どのCPUによって引き継ぐのかをCPU番号(各CPUを識別するためのシーケンシャルな番号)の系列で指定する遷移情報と、引き継ぐCPUが存在しない場合に、そのタスクだ

けを停止するか、システム全体を停止するかを指定する処理選択情報を持つ。

【0016】現在遷移位置テーブル4は、再構成テーブル2に記憶されている各タスクが現在それぞれのCPUで稼働しているかを、再構成テーブル2における遷移情報の系列の位置で示す現在遷移位置情報を保持する。

【0017】フォールトトレラント処理手段1は、他CPUに故障が生じた場合に、再構成テーブル2の遷移情報をもとに、その故障したCPU上で実行中のタスクが、自分(自CPU)が引き継ぐものかどうかを判断する引継判断部11と、そのタスクの処理を自CPUが引き継ぐ場合に、そのタスクの待機用タスク3を起動して、引き継ぎを行う待機タスク起動部12と、再構成テーブル2の現在遷移位置を自CPUを示す位置に更新する現在遷移位置更新部13とを備える。待機用タスク3は、あらかじめ生成しておいてもよく、また必要になったときに新たに生成して、制御を渡すことにより起動してもよい。

【0018】図2は、本発明の処理動作の流れの例を示す。この例では、CPU1～CPU4の番号を付与された4つのCPUからなるマルチプロセッサシステムにおいて、タスクIDが101のタスク(以下、タスク101と表記する)、タスクIDが201のタスク(以下、タスク201と表記する)、タスクIDが202のタスク(以下、タスク202と表記する)、タスクIDが301のタスク(以下、タスク301と表記する)の4つのタスクが実行されているとする。

【0019】再構成テーブル2には、あらかじめ各タスクごとに、タスクIDと障害発生時にそのタスクを引き継ぐCPU番号の順序を遷移情報として設定しておく。また、引き継ぐCPUがない場合にタスクだけを停止させるのか、システム全体として停止させるのかを示す処理選択の情報を設定しておく。タスク101のタスクを例に説明すると、再構成テーブル2には、それを実行しているCPUの障害時に、CPU1→CPU2→CPU3→CPU4の順に処理を引き継ぐことが設定され、引き継ぐCPUがない場合にはシステム全体の処理を停止するように設定されている。

【0020】現在遷移位置テーブル4には、再構成テーブル2に登録されている各タスクごとに、現在実行しているCPUが遷移情報中の先頭から何番目であるかを示す情報が、現在遷移位置として格納されている。故障しているCPUがない初期状態においては、現在遷移位置はすべて「1」である。すなわち、故障発生前では、タスク101はCPU1上で、タスク201、202はCPU2上で、タスク301はCPU3上で、それぞれ実行されていることが示されている。

【0021】CPU2が故障したとすると、それを検出したCPUから、CPU1、CPU3、CPU4は、CPU2の故障通知を受信する(ステップS1)。それぞ

れのCPUでは、再構成テーブル2を参照して、CPU2上で実行されていたタスク、すなわち遷移情報における1番目の現在遷移位置が「2(CPU2)」となっているタスクを探す(ステップS2)。ここで、CPU2上で実行されていたタスクが、タスク201、202であることがわかる。

【0022】CPU1は、タスク201の遷移情報(2→1)からこのタスクを引き継ぐCPUが自CPUであることがわかり、CPU4は、タスク202の遷移情報(2→4)からこのタスクを引き継ぐCPUが自CPUであることがわかる。そこで、CPU1は、201用タスクを起動させ、CPU4は、202用タスクを起動させる(ステップS3)。

【0023】CPU1、CPU4は、現在遷移位置テーブル4のタスク201、202の現在遷移位置をそれぞれ「1」から「2」に更新する(ステップS4)。

【0024】その後、CPU1、CPU3、CPU4が稼働している状態で、さらにCPU1が故障したとする。それを検出したCPUから、CPU3、CPU4は、CPU1の故障通知を受信する(ステップS5)。CPU3、CPU4は、現在遷移位置1、2、2、1をもとに、再構成テーブル2におけるタスク101、201、202、301の遷移情報のうち、それぞれ1番目、2番目、2番目、1番目を調べ、CPU1で実行されていたタスクがタスク101、201であることを知る。これらのタスク101、201の遷移情報から、引き継ぐCPU番号を得ると、タスク101については、引き継ぐべきCPU2が既に故障しているので、さらに次の遷移情報を調べて、次のCPU3が引き継ぐべきCPUであることがわかる。一方、タスク201は、引き継ぐべきCPUがないことがわかる(ステップS6)。

【0025】そこで、CPU3では101用タスクを起動させる(ステップS7)。引き継ぐべきCPUのないタスク201は、再構成テーブル2の処理選択情報を見ると「タスク」であるので、システム全体を停止させるのではなく、タスク201だけを停止させる。その後、現在遷移位置テーブル4におけるタスク101の現在遷移位置を「3」に更新し、引き継ぐべきCPUのないタスク201の現在遷移位置については、「なし」を示す「-1」に更新する(ステップS8)。

【0026】このように、本発明では、故障したCPU上で稼働していたタスクを引き継ぐCPUの引き継ぎ順序を、再構成テーブル2にCPUの番号を羅列して記すことにより、タスク単位でどのCPUが引き継ぐべきかを正常なCPUが最後の一つになるまで、容易に指定することができる。さらに、現在どのCPU上でタスクが実行されているのかを示すことで、次に引き継ぐCPUを簡単に調べることができる。また、引き継ぐCPUがなくなった場合の処理も簡単に設定することができ、タスクを停止させるだけで、他のタスクにより処理を続行

させるか、システム全体を停止させるかを容易に指定することができる。

【0027】このため、ある処理を実行するのに必要な通信回線やディスク記憶装置などのハードウェアを、すべてのCPUに同等に用意する必要がなく、タスクの処理機能に応じて特定のハードウェアが備わっているCPUにのみ確実に引き継がせることができるようになる。

【0028】図3は、本発明が利用される複数のCPUを搭載したシステムの例を示す。複数のCPUの接続形態としては、一般に知られているように、疎結合型と密結合型とがある。本発明はそのどちらの形態においても適用することができる。

【0029】図3(A)は、疎結合型の接続形態の例であって、各CPU20-1~20-nは、自己専用のプロセッサバス21-1~21-nとローカルメモリ22-1~22-nとを持ち、CPU20-1~20-n間で共有できるのはシステムバス23を通したグローバルメモリ24だけである。

【0030】図3(B)は密結合型の接続形態の例であって、各CPU30-1~30-mはプロセッサバス31もローカルメモリ32も共有し、ローカルメモリ32内を各CPU30-1~30-mがそれぞれ専用的に使うメモリ空間とCPU間で共有的に使うメモリ空間とを論理的に区分けして使用する。

【0031】図4に、疎結合型のマルチCPUシステムにおけるブロック構成例を示す。ここでは図を簡単に表すために、2台のCPUを示しているが、多数のCPUが接続されていてもよい。本発明に必要な各機能は、CPU20-1、20-1のローカルメモリ22-1、22-2に配置され、また、CPU間の共通資源を管理する領域はグローバルメモリ24に配置される。

【0032】図5に、密結合型のマルチCPUシステムにおけるブロック構成例を示す。本発明に必要な各手段は、CPU30-1、30-2ごとに、ローカルメモリ32の各CPU専用メモリ空間に配置され、また、CPU間の共通資源を管理する領域はローカルメモリ32内に設けられた共有メモリ空間に配置される。

【0033】図4および図5において、タスク管理部51は、自CPU上で動作するタスクを制御し、フォールトトレラント部53からタスクの引き継ぎ依頼があれば、引き継ぐために待機していたタスクを起動し、停止依頼があればタスクを停止する。故障検出部52は、自CPUまたは他のCPUの故障を検出し、CPU間通信部55を介して他のCPUに故障を通知する。フォールトトレラント部53は、他のCPUの故障を自ら検出するか、またはCPU間通信部55を介して、他のCPUの故障通知を受け取ると、再構成テーブル54を参照して、自CPUがタスクを引き継ぐかどうかを判断する。CPU間通信部55は、各CPU間でメッセージ通信を行うためのモジュールである。なお、図4に示すグロ-

バルメモリ24、または図5に示すローカルメモリ32内の共有メモリ空間には、CPU間の共通資源を管理するマルチCPU制御領域56が設けられる。また、この例では、現在遷移位置テーブル57も共有メモリ空間内に設けられている。

【0034】図6に、本発明をコンピュータによって実現させるためのプログラムの処理フローチャートを示す。このフローチャートは、特にフォールトトレラント部53を中心とした部分の処理の流れを示している。

【0035】まず、システムにおける一つのCPUの故障が検出された場合、再構成を行うために正常なCPU間で同期をとる(ステップS11)。故障の検出は、例えば各CPU間で定期的に生存確認のメッセージ交換を行うなど、従来から知られている方式を用いることができる。また、CPU間の同期は、共有メモリ空間におけるマルチCPU制御領域56またはCPU間通信部55を用いて行うが、同期方法については、よく知られているので、ここでの詳しい説明を省略する。

【0036】その後、再構成テーブル54の先頭のタスクから順番に、現在遷移位置テーブル57から得た現在遷移位置と、再構成テーブル54における遷移情報から、着目するタスクが現在どのCPU上で動作しているのかを調べる(ステップS12)。タスクが、既に動作していないタスクである場合、すなわち現在遷移位置テーブル57から得た現在遷移位置が「-1」の場合、ステップS21へ進む(ステップS13)。また、現在遷移位置の示すCPUが停止(故障)しているCPUでない場合、ステップS21へ進む(ステップS14)。

【0037】既に動作していないタスクではなく、また、そのタスクが割り当てられているCPUが停止しているCPUであった場合、再構成テーブル54の遷移情報から次にそのタスクを動作させるCPUを調べる(ステップS15)。

【0038】遷移情報に次のCPUが存在しない場合、ステップS23に進む(ステップS16)。遷移情報に次のCPUが存在するが、その次のCPUも停止しているときには(ステップS17)、ステップS15の処理へ戻り、再度、遷移情報から次にそのタスクを動作させるCPUを調べる。

【0039】また、次のCPUが停止しているCPUではない場合、そのタスクを引き継ぐCPUが自CPUであるかどうかを調べる(ステップS18)。自CPUである場合、ステップS19へ進み、自CPUでない場合、ステップS21へ進む。

【0040】タスクを引き継ぐCPUが自CPUである場合、現在遷移位置テーブル57の現在遷移位置を更新し(ステップS19)、待機させていたタスクを起動する(ステップS20)。

【0041】ステップS21では、再構成テーブル54におけるすべてのタスクについて、以上の処理を行った

かどうかを調べ、すべてのタスクについて以上の処理が終わるまで、ステップS12～S20を繰り返す。再構成テーブル54におけるすべてのタスクについて、以上の処理を行ったならば、他のCPUの処理を待ち、同期をとる(ステップS22)。正常のCPUのすべてにおいて、それぞれタスク引き継ぎ処理が完了し、同期がとれたならば、新しいタスク構成により業務処理を継続する。

【0042】また、ステップS15の処理において、遷移情報から次にそのタスクを動作させるCPUを調べ、次のCPUが存在しない場合(ステップS16)、そのタスクの処理選択情報を調べる(ステップS23)。処理選択情報がタスクであれば、現在遷移位置テーブル57の現在遷移位置を「-1」とし、そのタスクがシステムからなくなったことを記し(ステップS24)、その後、ステップS21へ進む。処理選択情報がタスクではなく、システム全体であれば、システム全体を停止させて処理を終了する(ステップS25)。

【0043】以上の実施の形態で説明したように、再構成テーブル54を各CPUのローカルメモリ32に配置し、現在遷移位置テーブル57を共有メモリ空間に配置することによって、再構成テーブル54へのアクセスの高速化と現在遷移位置の管理の容易化を実現することができ、好適なフォールトトレラント・システムを構築することができる。しかし、再構成テーブル54と現在遷移位置テーブル57とを、必ずしもローカルメモリ32と共有メモリ空間とに分けて配置しなければならないわけではなく、例えばローカルメモリ32または共有メモリ空間のいずれかにこれらのテーブルを共通に設けても、本発明を実施することができる。

【0044】

【実施例】次に、入力されたデータを加工して出力する以下のようなタスクを持つシステムを例にして、本発明の実施例の処理動作を説明する。このシステムは4つのCPU1～CPU4から構成されており、CPU1とCPU3には、入力装置を制御できるコントローラが配置され、CPU2とCPU4には出力装置を制御するコントローラが配置されているとする。

【0045】本システムで稼働するタスクは、

- ・入力処理タスク
- ・出力処理タスク
- ・データ加工マスタタスク
- ・データ加工サブタスク

である。

【0046】図7に、初期状態におけるタスクの構成例を示す。入力処理タスクはCPU1で、出力処理タスクはCPU2で稼働させる。データを加工する処理は分割して各CPUに分散させ、データ加工マスタタスクはCPU1に配置し、実際にデータを加工するデータ加工サブタスクは、すべてのCPUにそれぞれ配置して並列に

動作させる。

【0047】また、データ加工マスタタスクは、CPU2、CPU3、CPU4で待機させ、入力処理タスクはCPU3で、出力処理タスクはCPU4で、それぞれ待機させる。

【0048】図8に、再構成テーブルおよび現在遷移位置テーブルの設定例を示す。再構成テーブル54のタスクID、遷移情報、処理選択の情報を、図8に示すように設定して各CPUに配置する。再構成テーブル54は、あらかじめプログラム中に組み込んでおいてもよい。入力処理タスクについては、CPU1が故障した場合にはCPU3が引き継ぎ、引き継ぐCPUがない場合にはシステム全体の処理を停止するように設定されている。

【0049】出力処理タスクについては、CPU2が故障した場合にはCPU4が引き継ぎ、これも引き継ぐCPUがない場合にはシステム全体の処理を停止するように設定されている。データ加工マスタタスクは、稼働しているCPUが故障すると、CPU1、CPU2、CPU3、CPU4の順番で順次使用可能なCPUに処理が引き継がれ、使用可能なCPUがなくなると、システム全体の処理を停止するように設定されている。データ加工サブタスク1は、CPU1が故障しても他のCPUは引き継がず、そのタスク処理を停止するように設定されている。データ加工サブタスク2、3、4も、それぞれ動作しているCPU2、CPU3、CPU4が故障すると、そのタスクは引き継がれない。

【0050】現在遷移位置テーブル57は、共有メモリ空間の領域に割り当て、各現在遷移位置(各タスクが現在どのCPU上で稼働しているのかを示すための遷移情報の系列中の位置)は、システムの初期化時にすべて「1」に設定される。

【0051】今、CPU2に故障が発生し、CPU2を除いた構成でシステムを構成し直す場合、CPU1、CPU3、CPU4は、再構成テーブル54の遷移情報と現在遷移位置テーブル57の現在遷移位置とを参照して、CPU2上で稼働していたタスクを調べる。これにより、CPU2上で稼働していたタスクは、「出力処理タスク」と「データ加工サブタスク2」であること、および、出力処理タスクはCPU4が引き継ぎ、「データ加工サブタスク2」は引き継ぐ必要がないことがわかる。

【0052】遷移情報に従って、「データ加工サブタスク2」の現在遷移位置に「-1」を設定する。CPU4は、「出力処理タスク」を引き継ぐために待機させていたタスクを起動し、「出力処理タスク」がCPU4に引き継がれたことを示すために、現在遷移位置テーブル57における現在遷移位置を「2」に更新する。この状態のタスク構成は、図9に示すようになる。また、更新後の再構成テーブル54および現在遷移位置テーブル57

は、図10に示すようになる。

【0053】その後さらに、CPU1に故障が発生して、CPU1を除いてシステムを構成し直す場合、CPU3、CPU4は、現在遷移位置テーブル57の現在遷移位置と再構成テーブル54の遷移情報とを参照して、CPU1で稼働していたタスクを調べ、CPU1で稼働していたタスクが、「入力処理タスク」、「データ加工マスタタスク」、「データ加工サブタスク1」であること、および「入力処理タスク」はCPU3が引き継ぐこと、「データ加工サブタスク1」は引き継ぐ必要がないことがわかる。また、「データ加工マスタタスク」はCPU2が引き継ぐことがわかるが、CPU2は既に故障しているので、さらに次の遷移情報を調べて、CPU3が引き継ぐことを認識する。なお、CPU2が既に故障しているかどうかは、共有メモリ空間のマルチCPU制御領域56に記されているので、それによって認識する。

【0054】CPU3は、遷移情報から、「入力処理タスク」および「データ加工マスタタスク」を引き継ぐことを認識すると、これらを引き継ぐために待機させていたタスクをそれぞれ起動して、「入力処理タスク」の現在遷移位置を「2」に、「データ加工マスタタスク」の現在遷移位置を「3」に更新する。また、「データ加工サブタスク1」の現在遷移位置に「-1」を設定する。この状態のタスク構成は、図11に示すようになる。また、更新後の再構成テーブル54および現在遷移位置テーブル57は、図12に示すようになる。

【0055】

【発明の効果】以上説明したように、本発明によれば、マルチCPUシステムにおいて、故障したCPU上のタスクをどのCPUによって引き継ぐのかを、再構成テーブルにCPUの番号を羅列して記すことにより、容易に引き継ぎの順序を指定することができ、さらに、現在どのCPU上でタスクが実行されているのかを示すことで、次に引き継ぐCPUを簡単に調べることができる。

【0056】これにより、遷移情報ですべてのCPUを指定することによって、正常なCPUが最後の一つになるまで引き継ぐような指定や、特定のいくつかのCPUだけが引き継ぎを行うような指定を、簡単に設定することができる。

【0057】さらに、タスクごとの処理選択情報を再構成テーブル中に持たせることにより、引き継ぐCPUがなくなった場合にそのタスクだけを停止させたり、またはシステム全体を停止させたりするような指定も、簡単に設定することができる。

【0058】また、あらかじめ引き継ぐCPUを設定しておくため、タスクに必要なハードウェアを備えるCPUにだけ確実に引き継ぎを行わせることも可能になる。

【図面の簡単な説明】

【図1】本発明の概要を示すブロック構成図である。

【図2】本発明の処理動作の流れの例を示す図である。

【図3】マルチCPUシステムの接続形態の例を示す図である。

【図4】疎結合型システムにおけるブロック構成例を示す図である。

【図5】密結合型システムにおけるブロック構成例を示す図である。

【図6】本発明をコンピュータによって実現させるためのプログラムの処理フローチャートである。

【図7】本発明の実施例の初期状態におけるタスクの構成例を示す図である。

【図8】本発明の実施例の再構成テーブルおよび現在遷移位置テーブルの設定例を示す図である。

【図9】システムの各CPUのタスクの状態を説明するための図である。

【図10】更新後の再構成テーブルおよび現在遷移位置テーブルの例を示す図である。

【図11】システムの各CPUのタスクの状態を説明するための図である。

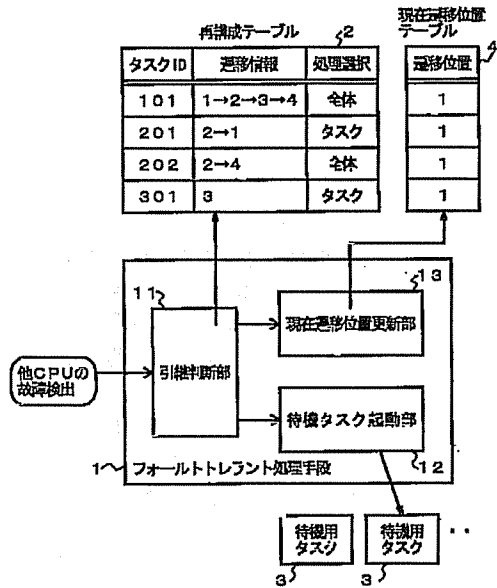
【図12】更新後の再構成テーブルおよび現在遷移位置テーブルの例を示す図である。

【符号の説明】

- 1 フォールトトレラント処理手段
- 11 引継判断部
- 12 待機タスク起動部
- 13 現在遷移位置更新部
- 2 再構成テーブル
- 3 待機用タスク
- 4 現在遷移位置テーブル

【図1】

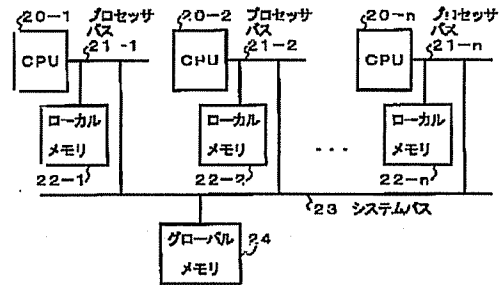
本発明のブロック構成図



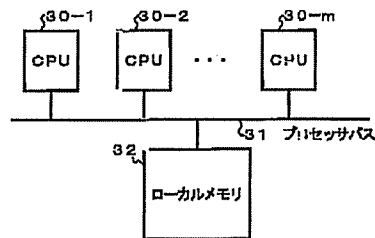
【図3】

接続形態の例

(A) 疎結合型

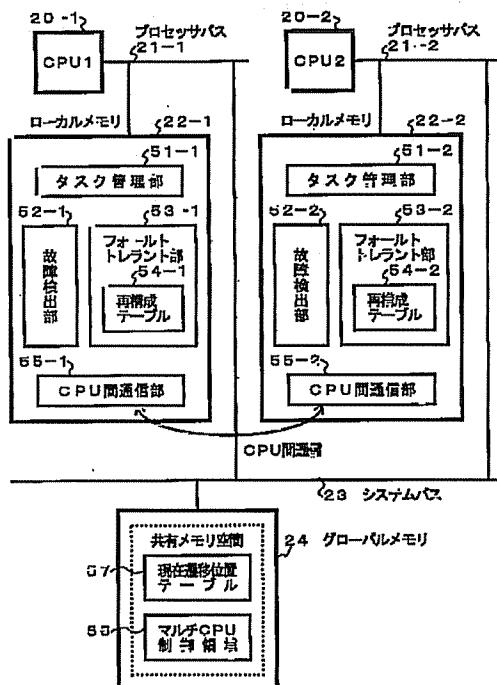


(B) 密結合型

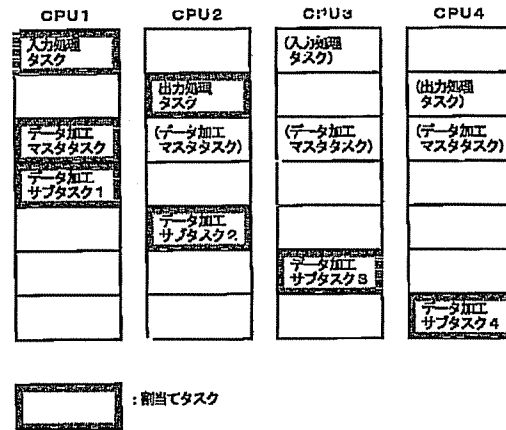


【図4】

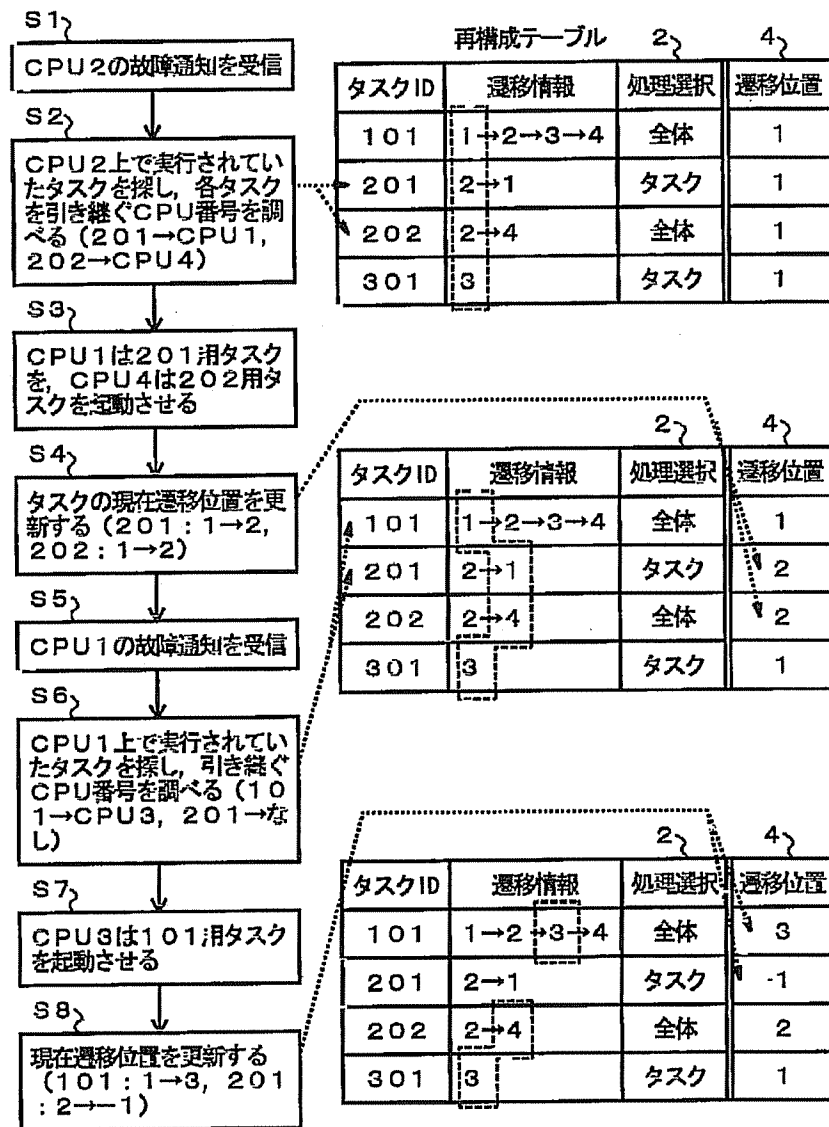
疎結合型システムにおけるブロック構成例



【図7】

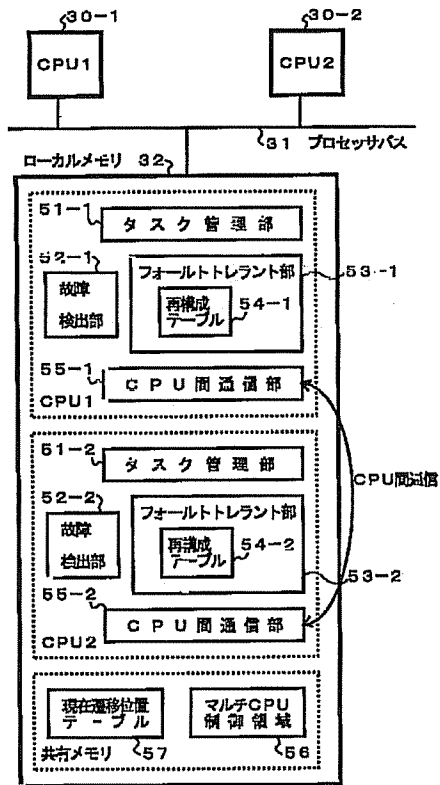


【図2】



【図5】

密結合型システムにおけるブロック構成例



【図8】

テーブルの設定例

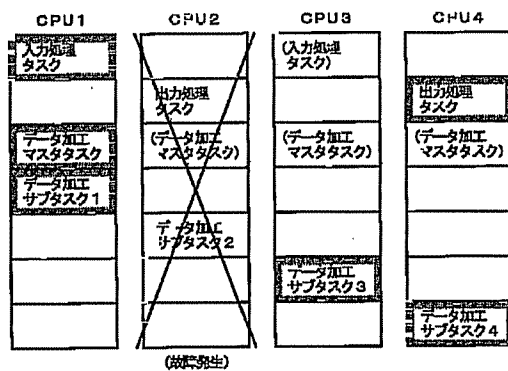
再構成テーブル 54	遷移情報	処理選択	現在遷移位置 テーブル 57
入力処理タスク	1→3	全体	1
出力処理タスク	2→4	全体	1
データ加工マスタタスク	1→2→3→4	全体	1
データ加工サブタスク1	1	タスク	1
データ加工サブタスク2	2	タスク	1
データ加工サブタスク3	3	タスク	1
データ加工サブタスク4	4	タスク	1

【図10】

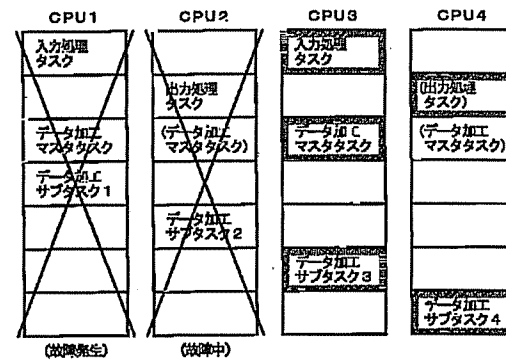
再構成テーブルおよび現在遷移位置テーブルの例

再構成テーブル 54	遷移情報	処理選択	現在遷移位置 テーブル 57
入力処理タスク	1→3	全体	1
出力処理タスク	2→4	全体	2
データ加工マスタタスク	1→2→3→4	全体	1
データ加工サブタスク1	1	タスク	1
データ加工サブタスク2	2	タスク	-1
データ加工サブタスク3	3	タスク	1
データ加工サブタスク4	4	タスク	1

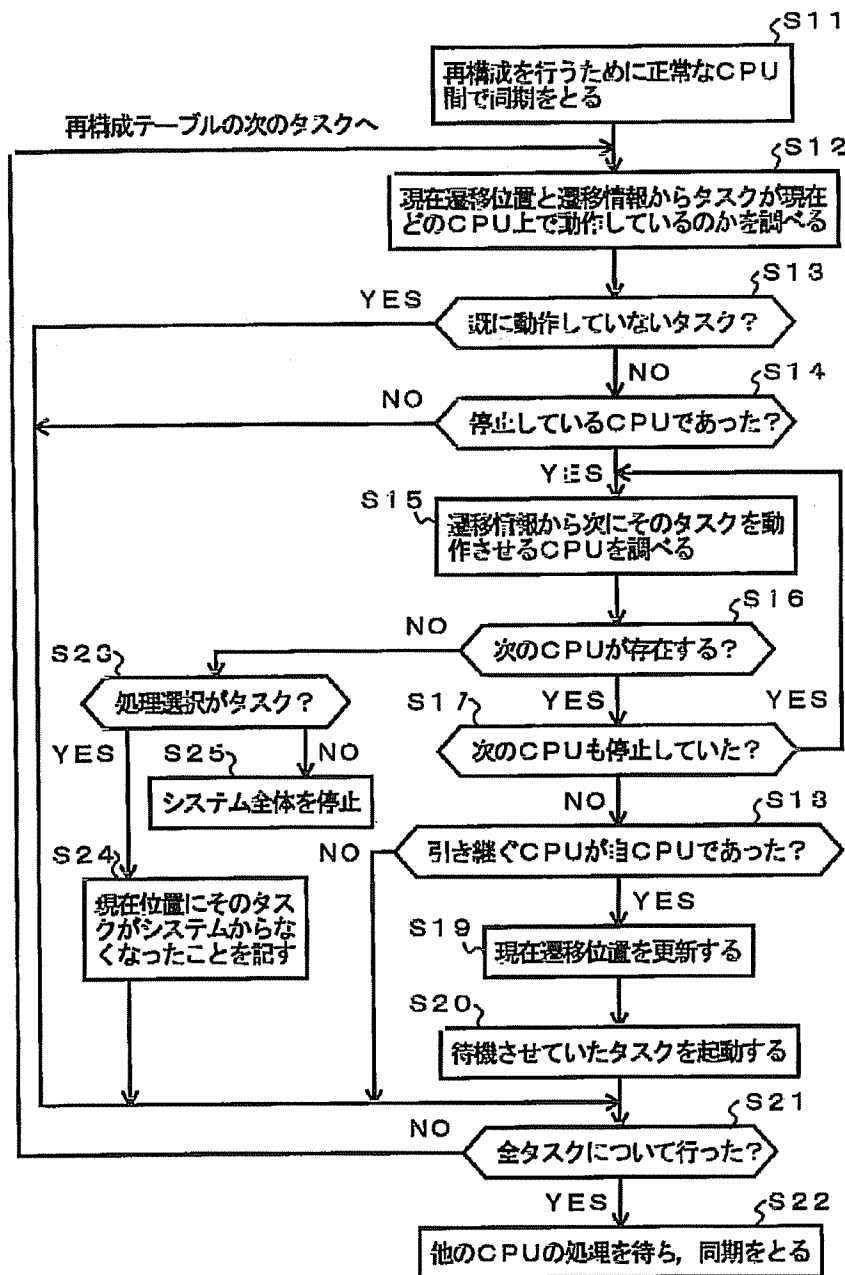
【図9】



【図11】



【図6】



【図12】

再構成テーブルおよび現在遷移位置テーブルの例

再構成テーブル 54			現在遷移位置 テーブル 57
タスク	遷移情報	処理選択	遷移位置
入力処理タスク	1→3	全体	2
出力処理タスク	2→4	全体	2
データ加工マスタタスク	1→2→3→4	全体	3
データ加工サブタスク1	1	タスク	-1
データ加工サブタスク2	2	タスク	-1
データ加工サブタスク3	3	タスク	1
データ加工サブタスク4	4	タスク	1

フロントページの続き

(72)発明者 岡本 二郎

神奈川県川崎市中原区上小田中4丁目1番
1号 富士通株式会社内

Fターム(参考) 5B034 BB11 CC01

5B045 BB02 BB12 GG06 JJ09 JJ13

JJ44

5B098 AA10 GA04 JJ00